

ПОтчет по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки

Паласиос Фелипе

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	12
	Список литературы	13

Список иллюстраций

3.1	программа	8
3.2	изменение рег есх	9
3.3	добавление команды push	9
3.4	аргументы	10
3.5	вывод суммы чисел	10
3.6	вычисление произведения аргументов	11
3.7	программа	11

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Создайте каталог для программ лабораторной работы No 8, перейдите в него и создайте файл lab8-1.asm:
2. Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работ
3. Измените текст программы добавив изменение значение регистра ecx в цикле:

```
label: sub ecx,1 ; ecx=ecx-1 mov [N],ecx mov eax,[N] call iprintLF
```

Создайте исполняемый файл и проверьте его работу.

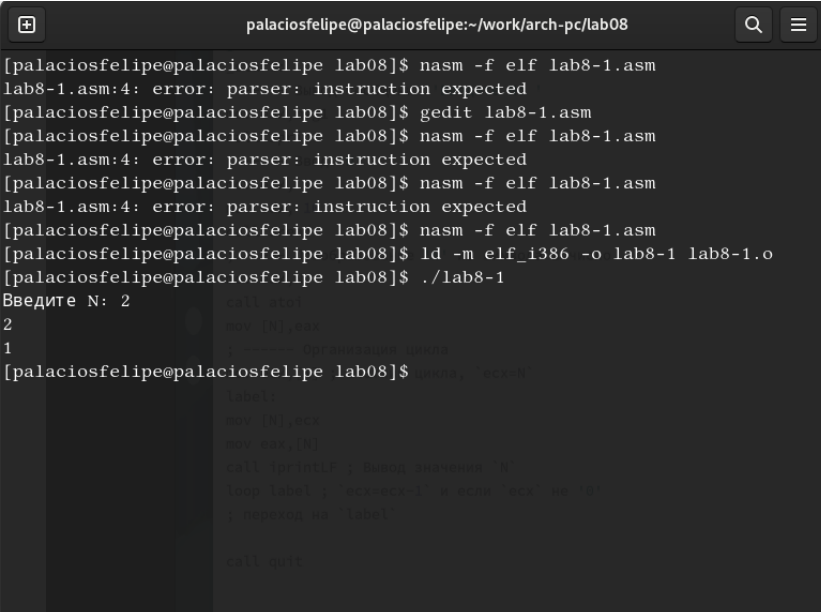
4. Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop: Создайте исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению ☒ введенному с клавиатуры?
5. Обработка аргументов командной строки Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст программы из листинга 8.2. Создайте исполняемый файл и запустите его, указав аргументы: Сколько аргументов было обработано программой?
6. Пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Создайте файл lab8-3.asm в каталоге ~/work/arch-

рс/lab08 и введите в него текст программы из листинга 8.3. Создайте исполняемый файл и запустите его, указав аргументы.

7. Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки
8. Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы.

3 Выполнение лабораторной работы

1. Создайте каталог для программ лабораторной работы No 8, перейдите в него и создайте файл lab8-1.asm:
2. Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работ (рис. 3.1).



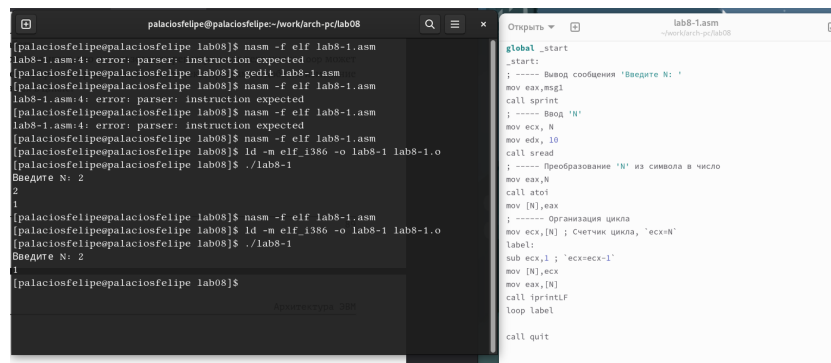
```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08
[palaciosfelipe@palaciosfelipe lab08]$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
[palaciosfelipe@palaciosfelipe lab08]$ gedit lab8-1.asm
[palaciosfelipe@palaciosfelipe lab08]$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
[palaciosfelipe@palaciosfelipe lab08]$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
[palaciosfelipe@palaciosfelipe lab08]$ nasm -f elf lab8-1.asm
[palaciosfelipe@palaciosfelipe lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[palaciosfelipe@palaciosfelipe lab08]$ ./lab8-1
Введите N: 2
2
1
[palaciosfelipe@palaciosfelipe lab08]$
```

Рис. 3.1: программа

3. Измените текст программы добавив изменение значение регистра ecx в цикле:

```
label: sub ecx,1 ; ecx=ecx-1 mov [N],ecx mov eax,[N] call iprintLF
```


Создайте исполняемый файл и проверьте его работу (рис. 3.2).



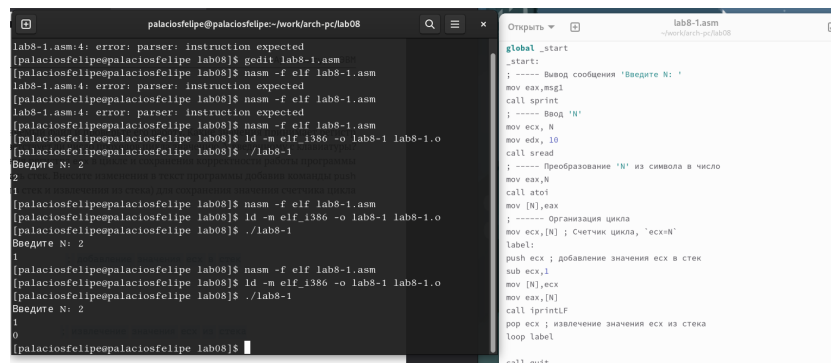
The image shows a terminal window on the left and an assembler window on the right. The terminal displays several error messages from the NASM assembler, indicating that the input file 'lab8-1.asm' is not found. The assembler window on the right shows the assembly code for 'lab8-1.asm', which includes instructions for reading input, converting it to a number, and organizing a loop.

```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ gedit lab8-1.asm
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 2
2
1
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 2
2
1
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$
```

```
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax, msg1
call sprintf
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax, N
call atoi
mov [N], eax
; ----- Организация цикла
mov ecx, [N]; Счетчик цикла, 'ecx=N'
label:
sub ecx, 1; 'ecx=ecx-1'
mov [N], ecx
mov eax, [N]
call fprintf
loop label
call quit
```

Рис. 3.2: изменение рег есх

- Для использования регистра есх в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop: Создайте исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению ☒ введенному с клавиатуры? (рис. 3.3).



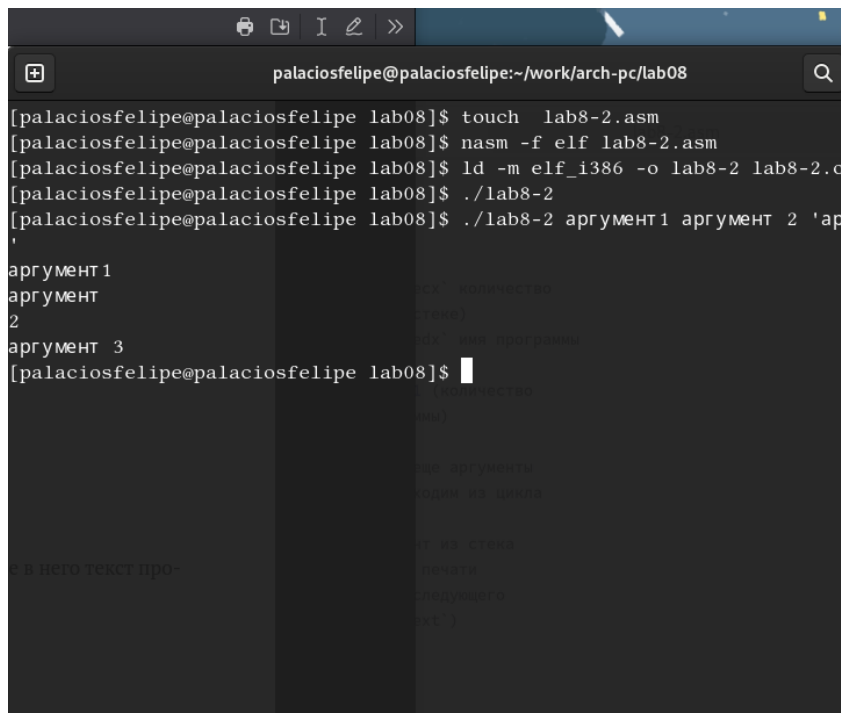
The image shows a terminal window on the left and an assembler window on the right. The terminal displays the same error messages as in Figure 3.2. The assembler window on the right shows the assembly code for 'lab8-1.asm', which includes instructions for reading input, converting it to a number, and organizing a loop using the stack to save the loop counter.

```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ gedit lab8-1.asm
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: parser: instruction expected
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 2
2
1
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 2
2
1
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08$
```

```
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax, msg1
call sprintf
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax, N
call atoi
mov [N], eax
; ----- Организация цикла
mov ecx, [N]; Счетчик цикла, 'ecx=N'
label:
push ecx; добавление значения ecx в стек
sub ecx, 1
mov [N], ecx
mov eax, [N]
call fprintf
pop ecx; извлечение значения ecx из стека
loop label
call quit
```

Рис. 3.3: добавление команды push

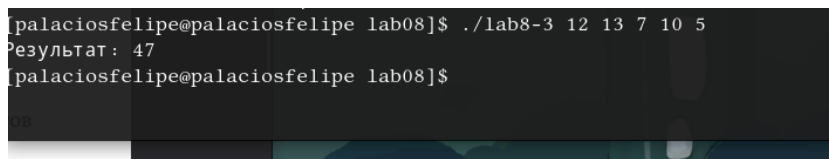
- Обработка аргументов командной строки Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст программы из листинга 8.2. Создайте исполняемый файл и запустите его, указав аргументы: Сколько аргументов было обработано программой? (рис. 3.4).



```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08
[palaciosfelipe@palaciosfelipe lab08]$ touch lab8-2.asm
[palaciosfelipe@palaciosfelipe lab08]$ nasm -f elf lab8-2.asm
[palaciosfelipe@palaciosfelipe lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[palaciosfelipe@palaciosfelipe lab08]$ ./lab8-2
[palaciosfelipe@palaciosfelipe lab08]$ ./lab8-2 аргумент1 аргумент 2 'аргумент3'
аргумент1
аргумент
2
аргумент 3
[palaciosfelipe@palaciosfelipe lab08]$
```

Рис. 3.4: аргументы

6. Пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Создайте файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст программы из листинга 8.3. Создайте исполняемый файл и запустите его, указав аргументы (рис. 3.5).



```
[palaciosfelipe@palaciosfelipe lab08]$ ./lab8-3 12 13 7 10 5
Результат: 47
[palaciosfelipe@palaciosfelipe lab08]$
```

Рис. 3.5: вывод суммы чисел

7. Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 3.6).

The image shows a code editor with two tabs: 'lab8-1.asm' and 'lab8-2.asm'. The 'lab8-2.asm' tab is active, displaying assembly code for a function that calculates the product of its arguments. The code includes sections for data (result), text (loop), and global _start. It uses instructions like pop, sub, mov, cmp, jz, call, mul, and loop. A terminal window on the right shows the execution of the program using nasm and ld, followed by running the resulting binary with arguments 12, 13, 7, 10, and 5, resulting in an output of 47.

```

lab8-2.asm
~\work\arch-pc\lab08

%include "in_out.asm"
SECTION .data
msg db 'результат: '
SECTION .text
GLOBAL _start

_start:
pop ecx
pop edx
sub ecx,1
mov esi,1

next:
cmp ecx,0
jz _end

pop eax
call atoi
mul esi
mov esi, eax

loop next

_end:
mov eax, msg

```

```

palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08
[palaciosfelipe@palaciosfelipe lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[palaciosfelipe@palaciosfelipe lab08]$ ./lab8-3
результат: 0
[palaciosfelipe@palaciosfelipe lab08]$ ./main 12 13 7 10 5
bash: ./main: Нет такого файла или каталога
[palaciosfelipe@palaciosfelipe lab08]$ ./main 12 13 7 10 5
bash: ./main: Нет такого файла или каталога
[palaciosfelipe@palaciosfelipe lab08]$ ./lab8-3 12 13 7 10 5
результат: 47
[palaciosfelipe@palaciosfelipe lab08]$ nasm -f elf lab8-3.asm
[palaciosfelipe@palaciosfelipe lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[palaciosfelipe@palaciosfelipe lab08]$ ./lab8-3
результат: 1
[palaciosfelipe@palaciosfelipe lab08]$ ./lab8-3 1 2
результат: 2
[palaciosfelipe@palaciosfelipe lab08]$

```

Рис. 3.6: вычисление произведения аргументов

8. Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы (рис. 3.7).

The image shows a code editor with a tab 'lab8-f.asm'. The code defines a function f(x) that calculates 10(x-1) and a main function that sums the results of f(x) for arguments 10, 13, and 7. The code includes sections for data (result, msg), text (loop), and global _start. It uses instructions like pop, sub, mov, cmp, jz, call, dec, mul, add, loop, and call. A terminal window on the right shows the execution of the program using nasm and ld, followed by running the resulting binary with arguments 10, 13, and 7, resulting in an output of 0.

```

lab8-f.asm
~\work\arch-pc\lab08

1 %include "in_out.asm"
2
3 SECTION .data
4 f_x db "функция: 10(x - 1)",0h
5 msg db "10,13,7,результат: ",0h
6
7 SECTION .text
8 global _start
9
10 _start:
11 pop ecx
12 pop edx
13 sub ecx, 1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 dec eax
22 mov ebx, 10
23 mul ebx
24 add esi, eax
25
26 loop next
27
28 _end:
29 mov eax, f_x
30 call sprint
31 mov eax, msg
32 call sprint
33 mov eax, esi
34 call printf
35
36 call quit

```

```

palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab08
[palaciosfelipe@palaciosfelipe lab08]$ nasm -f elf lab8-f.asm
[palaciosfelipe@palaciosfelipe lab08]$ ld -m elf_i386 -o lab8-f lab8-f.o
[palaciosfelipe@palaciosfelipe lab08]$ ./lab8-f
функция: 10(x - 1)
результат: 0
[palaciosfelipe@palaciosfelipe lab08]$

```

Рис. 3.7: программа

4 Выводы

Приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы